Working with Universal Dependencies

Martin Popel
popel@ufal.mff.cuni.cz

ÚFAL (Institute of Formal and Applied Linguistics)
Charles University, 2017-03-13

"I SPEND A LOT OF TIME ON THIS TASK. I SHOULD WRITE A PROGRAM AUTOMATING IT!"

Diagram:

Theory:
- Work
- Writing code
- Work on original task
- Automation takes over
- Free time

Reality:
- Work
- Writing code
- Debugging
- Rethinking
- Ongoing development
- No time for original task anymore

https://xkcd.com/1319/
Overview

- Universal Dependencies
- A thought on simplicity
- CoNLL 2017 Shared Task
- Tools for UD
- Udapi
Universal Dependencies

- cross-linguistically consistent treebank annotation
- de facto standard for dependency annotation
- builds on:
  Stanford Dependencies, CoNLL, Google UPOS + UDT, HamleDT, Interset
- balances:
  details vs. simplicity (linguists vs. NLP applications)
  adequacy for a given lang. vs. cross-ling
- started in 2014 (kick-off, guidelines v1)
- first release in 2015, new release each 6 months
UD releases

UDv2.0 released on 2017-03-01:
145 contributors, 70 treebanks, 50 langs, 12M words

UD popularity wrt. academic papers

457 papers about Universal Dependencies, 582 citations

(source: Google Scholar, March 2017)
Universal Dependencies v2

Executive summary of changes from v1 to v2

- Tokenization and word segmentation
- Morphology
  - General principles
  - Universal POS tags (single document)
  - Universal features (single document)
  - Language-specific features
  - Conversion from other tagsets
- Syntax
  - General principles
  - Basic dependencies
    - Simple clauses
    - Nominals
    - Complex clauses
    - Other constructions
  - Enhanced dependencies
  - Universal dependency relations (single document)
  - Language-specific relations
- CoNLL-U format
### Universal POS tags

<table>
<thead>
<tr>
<th>Open class words</th>
<th>Closed class words</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADJ</td>
<td>ADP</td>
<td>PUNCT</td>
</tr>
<tr>
<td>ADV</td>
<td>AUX</td>
<td>SYM</td>
</tr>
<tr>
<td>INTJ</td>
<td>CCONJ</td>
<td></td>
</tr>
<tr>
<td>NOUN</td>
<td>DET</td>
<td></td>
</tr>
<tr>
<td>PROPN</td>
<td>NUM</td>
<td>X</td>
</tr>
<tr>
<td>VERB</td>
<td>PART</td>
<td></td>
</tr>
</tbody>
</table>
### Universal features

<table>
<thead>
<tr>
<th>Lexical features</th>
<th>Inflectional features</th>
</tr>
</thead>
<tbody>
<tr>
<td>PronType</td>
<td>Gender</td>
</tr>
<tr>
<td>NumType</td>
<td>Animacy</td>
</tr>
<tr>
<td>Poss</td>
<td>Number</td>
</tr>
<tr>
<td>Reflex</td>
<td>Case</td>
</tr>
<tr>
<td>Foreign</td>
<td>Definite</td>
</tr>
<tr>
<td>Abbr</td>
<td>Degree</td>
</tr>
<tr>
<td></td>
<td>Polarity</td>
</tr>
<tr>
<td></td>
<td>Person</td>
</tr>
<tr>
<td></td>
<td>Polite</td>
</tr>
</tbody>
</table>

**Nominal\(^*\)**
- Gender
- Animacy
- Number
- Case
- Definite
- Degree
- Polarity
- Person
- Polite

**Verbal\(^*\)**
- VerbForm
- Mood
- Tense
- Aspect
- Voice
- Evident

**Index:**
- abbreviation, abessive, ablative, absolute superlative, absolutive, accusative, active, additive, adessive, admirative, adverbial participle, affirmative, allative, animate, antipassive, aorist, article, aspect, associative, benefactive, cardinal, case, causative case, causative voice, collective noun, collective numeral, collective pronominal, comitative, common gender, comparative case, comparative degree, complex definiteness, conditional, conjunctive, construct state, convert, count plural, counting form, dative, definite, definiteness, degree of comparison, delative, demonstrative, desiderative, destinative, direct case, direct voice, directional allative, distributive case, distributive numeral, dual, elative, elevated referent, emphatic, equative
## Deprels (dependency relations)

<table>
<thead>
<tr>
<th>Core arguments</th>
<th>Nominals</th>
<th>Clauses</th>
<th>Modifier words</th>
<th>Function Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>nsubj</td>
<td>csubj</td>
<td>advmod*</td>
<td>aux</td>
<td></td>
</tr>
<tr>
<td>obj</td>
<td>ccomp</td>
<td>discourse</td>
<td>cop</td>
<td></td>
</tr>
<tr>
<td>iobj</td>
<td>xcomp</td>
<td></td>
<td>mark</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-core dependents</th>
<th>Nominal dependents</th>
<th>Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>obl</td>
<td>nmod</td>
<td>MWE</td>
</tr>
<tr>
<td>vocative</td>
<td>appos</td>
<td>fixed</td>
</tr>
<tr>
<td>expl</td>
<td>nummod</td>
<td>flat</td>
</tr>
<tr>
<td>dislocated</td>
<td></td>
<td>compound</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coordination</th>
<th>MWE</th>
<th>Loose</th>
<th>Special</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>conj</td>
<td>list</td>
<td>orphan</td>
<td>punct</td>
<td></td>
</tr>
<tr>
<td>cc</td>
<td>parataxis</td>
<td>goeswith</td>
<td>root</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>reparandum</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
UD v2 guidelines changes

- spaces in forms and lemmas (Vietnamese, “100 000”, “i.e.”)
- few tags, deprels and features renamed (CONJ→CCONJ, dobj→obj, Negative→Polarity, ...)
- new deprels (obl, clf), features (Polite, ...) and values
- removed deprels (neg, auxpass)
- copula verbs tagged as AUX

![Diagram of dependency relations between words](image)
UD v2 guidelines changes

- coordinating conjunction & punctuation attached to the next conjunct (not the first conjunct)

UDv1:

UDv2:
remnant-style ellipsis → orphan-style

UDv1:

UDv2:
UD v2 guidelines changes

- empty nodes allowed in enhanced deps

**basic:**
```
I like tea and you rum .
```

**enhanced:**
```
I like tea and you E5.1 rum .
```
UD v2 guidelines changes

- empty nodes allowed in enhanced deps
- DEPS column with all enhanced deps (not just the extra)

1. I  I  PRON  PRP  Number=Sing|Person=1|PronType=Prs  2  nsubj  2:nsubj _
2. like  like  VERB  VBP  Mood=Ind|Tense=Pres|VerbForm=Fin  0  root  0:root _
3. tea  tea  NOUN  NN  Number=Sing
4. and  and  CCONJ  CC  _
5. you  you  PRON  PRP  Case=Nom|Person=2|PronType=Prs  2  conj  5.1:nsubj _
5.1  _  _  VERB  VBP  Mood=Ind|Tense=Pres|VerbForm=Fin  _  _  2:conj _
6. rum  rum  NOUN  NN  Number=Sing
7. .  .  PUNCT  .  _
UD v2 guidelines changes

- 4 types of enhanced dependencies specified
  - controlled/raised subjects
  - ellipsis
  - propagation of conjunct
  - relative clauses
  - case information

![Basic Diagram](image1)

- Mary wants to buy a book.

![Enhanced Diagram](image2)

- Mary wants to buy a book.

- She seems to be reading a book.
4 types of enhanced dependencies specified
- controlled/raised subjects
- ellipsis
- propagation of conjunct
- relative clauses
- case information
UD v2 guidelines changes

4 types of enhanced dependencies specified
- controlled/raised subjects
- ellipsis
- propagation of conjunct
- relative clauses
- case information
UD v2 guidelines changes

- 4 types of enhanced dependencies specified
  - controlled/raised subjects
  - ellipsis
  - propagation of conjunct
  - relative clauses
  - case information

```
the house on the hill
He went to the dinner after leaving work
```

```
DET# die NOUN# Zerstörung DET# der NOUN# Stadt
```

```
DET# die NOUN# Zerstörung der Stadt
```
UD v2 guidelines changes

- sent_id and text comments required (and SpaceAfter=No)

```
# sent_id = 1
# text = I like tea and you rum.
1 I I PRON PRP Number=Sing | Person=1 | PronType=Prs 2 nsubj 2:nsubj _
2 like like VERB VBP Mood=Ind | Tense=Pres | VerbForm=Fin 0 root 0:root _
3 tea tea NOUN NN Number=Sing 2 obj 2:obj _
4 and and CCONJ CC _ 5 cc 5.1:cc _
5 you you PRON PRP Case=Nom | Person=2 | PronType=Prs 2 conj 5.1:nsubj_ 5.1:__.
5.1 _ _ VERB VBP Mood=Ind | Tense=Pres | VerbForm=Fin _ _ _ 2:conj _
6 rum rum NOUN NN Number=Sing 5 orphan 5.1:obj SpaceAfter=No
7 . . PUNCT . _ 2 punct 2:punct _
```
How simple is simple enough?

**TYPICAL APPLE PRODUCT...**

![Touch interface](image)

**A GOOGLE PRODUCT...**

![FIND interface](image)

**YOUR COMPANY’S APP...**

![Complex interface with various fields](image)
File format: CoNLL-U (1 file, 6 lines, 218 bytes)

<table>
<thead>
<tr>
<th>sent_id = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>text = John loves Mary</td>
</tr>
<tr>
<td>1 John John PROPN NNP Number=Sing</td>
</tr>
<tr>
<td>2 loves love VERB VBZ Mood=Ind</td>
</tr>
<tr>
<td>3 Mary Mary PROPN NNP Number=Sing</td>
</tr>
</tbody>
</table>
<?xml version="1.0" encoding="UTF-8"?>
<treex_document xmlns="http://ufal.mff.cuni.cz/pdt/pml/">
    <head>
        <schema href="treex_schema.xml"/>
    </head>
    <meta/>
    <bundles>
        <LM id="1">
            <zones>
                <zone language="und">
                    <sentence>John loves Mary</sentence>
                    <trees>
                        <a_tree id="1/und">
                            <ord>0</ord>
                            <children id="n2">
                                <children>
                                    <LM id="n1">
                                        <form>John</form>
                                        <lemma>John</lemma>
                                        <tag>PROPN</tag>
                                        <iset>
                                            <pos>noun</pos>
                                            <nountype>prop</nountype>
                                            <number>sing</number>
                                        </iset>
                                        <ord>1</ord>
                                        <deprel>subj</deprel>
                                        <conll>
                                            <deprel>subj</deprel>
                                            <cpos>PROPN</cpos>
                                            <pos>NNP</pos>
                                            <feat>Number=Sing</feat>
                                        </conll>
                                    </LM>
                                    <LM id="n3">
                                        <form>Mary</form>
                                        <lemma>Mary</lemma>
                                        <tag>PROPN</tag>
                                        <iset>
                                            <pos>noun</pos>
                                            <nountype>prop</nountype>
                                            <number>sing</number>
                                        </iset>
                                    </LM>
                                </children>
                            </children>
                        </a_tree>
                    </trees>
                </zone>
            </zones>
        </LM>
    </bundles>
</treex_document>
John loves Mary.

(no morpho features)
Prague Markup Language (PML)

Quotes from the newest documentation:

- PML is a common basis of an open family of XML-based data formats.
- PML is an on-going project in its early stage.

Facts:

- PML is used in Prague-* treebanks (and lexicons), Tamil, Indonesian, Latin and Ancient Greek DT, Lithuanian DT,...
- PML has just one implementation (Perl): Fslib/TrEd/Treex::PML
- PML is difficult to understand and maintain.
Guidelines (number of pages) – Let’s compare uncomparable...

<table>
<thead>
<tr>
<th></th>
<th>UD</th>
<th>PDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>intro</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>morpho</td>
<td>61</td>
<td>56</td>
</tr>
<tr>
<td>syntax</td>
<td>60</td>
<td>317</td>
</tr>
<tr>
<td>tecto</td>
<td>–</td>
<td>1287</td>
</tr>
<tr>
<td>format</td>
<td>7</td>
<td>22</td>
</tr>
<tr>
<td>schema</td>
<td>–</td>
<td>59</td>
</tr>
<tr>
<td>total</td>
<td>131</td>
<td>1794</td>
</tr>
</tbody>
</table>

### UD 2.0 & CoNLL-U vs. PDT 2.0 & PML

**Guidelines (number of pages) – Let’s compare incomparable...**

<table>
<thead>
<tr>
<th></th>
<th>UD</th>
<th>PDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>intro</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>morpho</td>
<td>61</td>
<td>56</td>
</tr>
<tr>
<td>syntax</td>
<td>60</td>
<td>317</td>
</tr>
<tr>
<td>tecto</td>
<td>–</td>
<td>1287</td>
</tr>
<tr>
<td>format</td>
<td>7</td>
<td>22</td>
</tr>
<tr>
<td>schema</td>
<td>–</td>
<td>59</td>
</tr>
<tr>
<td>total</td>
<td>131</td>
<td>1794</td>
</tr>
</tbody>
</table>

Guide “to become quickly familiar with the basic ideas”

**UD 2.0 & CoNLL-U vs. PDT 2.0 & PML**

**Guidelines (number of pages)** – Let’s compare incomparable…

<table>
<thead>
<tr>
<th></th>
<th>UD</th>
<th>PDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>intro</td>
<td>3</td>
<td>53</td>
</tr>
<tr>
<td>morpho</td>
<td>61</td>
<td>56</td>
</tr>
<tr>
<td>syntax</td>
<td>60</td>
<td>317</td>
</tr>
<tr>
<td>tecto</td>
<td>–</td>
<td>1287</td>
</tr>
<tr>
<td>format</td>
<td>7</td>
<td>22</td>
</tr>
<tr>
<td>schema</td>
<td>–</td>
<td>59</td>
</tr>
<tr>
<td>total</td>
<td>131</td>
<td>1794</td>
</tr>
</tbody>
</table>


*guide “to become quickly familiar with the basic ideas”*

*“PML also tries to retain simplicity.”*
Simplicity?

- Simplicity is not only about file size or doc pages.
- Success is not only about simplicity.
- UD/CoNLL-U approach:
  *Let simple things be ultra-simple*
  (at the cost of edge cases being less elegant or not possible).
  - literal underscore token in CoNLL-U
  - original text including inter-sentence spaces
  - add new type of annotation
    (cross-sentence coreference, alignment)
Simplicity: Coordinations

PDT

a-tree
zone=und

and
Coord

store
Sb

buys
Pred_Co

The
AuxA

UD basic

a-tree
zone=und

buys
root

store
insubj

sells
conj

The
det

and
cc

cameras
obj

cameras
Obj
Simplicity: Coordinations

PDT

UD basic

UD enhanced
CoNLL 2017 multilingual parsing shared task

- http://universaldependencies.org/conll17/
- from Raw Text to Universal Dependencies
- segmentation, tokenization, labelled parsing (no morpho)
- 45 UDv2 languages plus X surprise languages
- registration deadline: April 15
- test phase: May 8–12
- organizers: ÚFAL, Google, Uppsala, Turku
- participants: 72 teams so far
Tools for annotating dependency trees

- **TrEd (+EasyTreex):** powerful, customizable, Perl, old
- **Brat:** online/JS+Python, UD support, embeddable
- **EasyTree:** perhaps too simple
- **GraphAnno:** useful for discourse etc., Java, keyboard
Tools for annotating dependency trees

- **TrEd (+EasyTreex):** powerful, customizable, Perl, old
- **Brat:** online/JS+Python, UD support, embeddable
- **EasyTree:** perhaps too simple
- **GraphAnno:** useful for discourse etc., Java, keyboard
Tools for annotating dependency trees

- **TrEd (+EasyTreex):** powerful, customizable, Perl, old
- **Brat:** online/JS+Python, UD support, embeddable
- **EasyTree:** perhaps too simple
- **GraphAnno:** useful for discourse etc., Java, keyboard
Tools for annotating dependency trees

- **TrEd (+EasyTreex):** powerful, customizable, Perl, old
- **Brat:** online/JS+Python, UD support, embeddable
- **EasyTree:** perhaps too simple
- **GraphAnno:** useful for discourse etc., Java, keyboard
**Automatic parsing: UDPipe**

UDPipe by Milan Straka, try it online/as a webservice
http://lindat.mff.cuni.cz/services/udpipe/

- End-to-end, batteries included:
  - segment, tokenize, tag, morpho, lemma, labelled parsing
- Pretrained models for all the UD (2.0 soon) langs
- User friendly (outputs CoNLL-U, Table, SVG)
- State-of-the-art quality, ultra fast
- Open-source, easy install for Linux, OS X, Win
- Interfaces for C++, C#, Java, Perl, Python
- Easily train on your own data
Tools for viewing dependency trees

- all the editors, including UDPipe online
- PML-TQ (UDv1.2)
  https://lindat.mff.cuni.cz/services/pmltq/
- Udapi https://github.com/udapi/udapi-python
  udapy write.Html < my.conllu > my.html
Tools for viewing dependency trees


zone=cs
id=f000001-s1/cs

jsem
VERB
root

Kde
ADV
advmod

.punct
PUNCT

zone=en
id=f000001-s1/en

am
VERB
root

Where
ADV
dep

I
PRON
nsubj

VerbForm=Fin

Save as SVG
Tools for viewing dependency trees

- all the editors, including UDPipe online
- PML-TQ (UDv1.2)  
  https://lindat.mff.cuni.cz/services/pmltq/
- Udapi  https://github.com/udapi/udapi-python
  
  udapy write.Html < my.conllu > my.html  
  demo: http://ufallab.ms.mff.cuni.cz/~popel/czeng1.6-sample.html
  
  udapy -HA < my.conllu > my.html
Tools for viewing dependency trees
Tools for viewing dependency trees

- all the editors, including UDPipe online
- PML-TQ (UDv1.2)
  https://lindat.mff.cuni.cz/services/pmltq/
- Udapi https://github.com/udapi/udapi-python

udapy write.Html < my.conllu > my.html
demo:  http://ufallab.ms.mff.cuni.cz/~popel/czeng1.6-sample.html

udapy -HA < my.conllu > my.html
udapy -T < my.conllu | less -R
Tools for viewing dependency trees

# sent_id = 1
# text = Corriere Sport da pagina 23 a pagina 26

Corriere PROPN root
  Sport PROPN name
    da ADP case
    pagina NOUN nmod
      23 NUM nummod
      a ADP case
      pagina NOUN nmod
      26 NUM nummod

# sent_id = 2
# text = I tre avevano da poco lasciato la cima e stavano cominciando la discesa.

I DET det
  tre NUM nsubj
  avevano AUX aux
    da ADP case
    poco ADV advmod
  lasciato VERB root
    la DET det
    cima NOUN dobj
    e CONJ cc
    stavano AUX aux
    cominciando VERB conj
      la DET det
      discesa NOUN dobj
      . PUNCT punct
Tools for viewing dependency trees

- all the editors, including UDPipe online
- PML-TQ (UDv1.2)
  https://lindat.mff.cuni.cz/services/pmltq/
- Udapi https://github.com/udapi/udapi-python

```
udapy write.Html < my.conllu > my.html
demo: http://ufallab.ms.mff.cuni.cz/~popel/czeng1.6-sample.html

udapy -HA < my.conllu > my.html

udapy -T < my.conllu | less -R
T=text, H=html, A=all attributes, N=no color, M=marked
```
Udapi

- [http://udapi.github.io/](http://udapi.github.io/)
- API and multi-language framework for processing UD
- Allows both fast prototyping and full applications
- Both command-line tool (udapy) and library
- Modularity, reusability, cooperation
Udapi use cases

- format conversions (CoNLL-U, SDParse, PML, VISL-cg, TikZ)
- `ud.Convert1to2` transformations from UD v1 to v2
  used for Bulgarian, Romanian, Galician, Russian, Irish, ...
- `ud.MarkBugs` validity tests
- `ud.SetSpaceAfter, ud.SetSpaceAfterFromText`
- `util.Eval, util.Filter, util.Wc`
- automatic parsing (via UDPipe), evaluation,...

Hands-on tutorial
http://udapi.github.io/tutorial/
Treex vs. Udapi

**Treex (2005)**
- Perl
- over 10 ÚFAL developers
- multilingual in the end
- Prague-style dependencies
- tectogrammatical layer
- PML / XML / *.treex
- weak UD support
- focus: TectoMT

[https://github.com/ufal/treex](https://github.com/ufal/treex)

**Udapi (2016)**
- Python, Perl, Java
- MP+ZŽ+M.Vojtek
- multilingual from the start
- Universal dependencies
- no layers (but zones)
- CoNLL-U
- MWT, empty, enhanced
- focus: speed & simplicity

[https://github.com/udapi/](https://github.com/udapi/)
Benchmark: Speed-up relative to Treex

(source: https://github.com/martinpopel/newtreex)
Benchmarked: Memory (MB)

(source: https://github.com/martinpopel/newtreex)

cs-ud-train-l.conllu: 68 MiB, 41k sentences, 0.8 MWords
Algorithmic challenges

- data structure for globally-ordered rooted trees
  
  node.descendants ... ordered
  
  node.shift_before_node()

- efficient loading & saving of CoNLL-U files
  
  linear-time checking of cycles
  
  lazy deserialization of FEATS and MISC

- write.TextModeTrees for non-projective trees
  
  minimize crossings and/or depth

- align raw sentence with gold-annotated words
  
  add SpaceAfter=No and goeswith
  
  create multi-word tokens (minimal)
  
  mark typos etc. in MISC
Summary

```
EVERY NOW AND THEN I REALIZE I'M MAINTAINING A
HUGE CHAIN OF TECHNOLOGY SOLELY TO SUPPORT ITSELF.
```

https://www.xkcd.com/1579/